FOCUS OF ATTENTION IN DECISION SUPPORT SYSTEMS

John O. Gurney, Jr.
Army Research Laboratory
Adelphi, MD

Elaine Marsh and Kenneth Wauchope
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory
Washington, DC 20375-5337

1.0 Introduction – Natural Language in Decision Support

In this paper, we argue that natural language interfaces with discourse capabilities have an important role to play in decision support systems, because such interfaces are responsive to focus of attention in the mind of a human decision maker. The decision support systems that we will discuss employ graphical user interfaces. They are characterized by strict ordering requirements for commands, no reference to previous operations, and an often arbitrary, redundant, and repetitive order of operations. Natural language interfaces (NLIs) allow English language commands and questions to be input either via a keyboard or using a speech input device in a relatively natural and easy way, and discourse capabilities allow a user to input commands or questions that are connected in some specifiable ways. We will provide examples from systems that have been developed at the Army Research Laboratory (ARL) and at the Naval Research Laboratory (NRL). Currently these two laboratories are cooperating under the JDL Centers of Excellence in Artificial Intelligence Program in which NRL operates as a Center of Excellence for Natural Language Processing. We believe that research on natural language discourse will improve the capabilities of natural language interfaces to decision support systems as well as provide extensions that can ultimately make their graphical interfaces easier and more natural to use [Marsh 1991].

We will consider interfaces developed at NRL and ARL for three decision support systems. In recent years the Interactive Systems group at NRL's Navy Center for Applied Research in Artificial Intelligence has been developing natural language (NL) interfaces for existing decision, planning, and training tools. One such tool is the KOALAS Test Planning Tool (here called simply KOALAS for short), which simulates scenario-generated engagements between friendly and hostile fighter aircraft. At ARL another natural language interface has been prototyped for the Combat Information Processor (CIP). The CIP integrates, processes, and displays battlefield information for use by intelligence, and command and control officers. A third system developed in the Advanced Information Technology Branch at NRL is TABS, an antisubmarine battle management system.

The user interfaces of the CIP, KOALAS, and TABS are remarkably similar in style. Each of these systems uses a map displayed on a Sun workstation monitor to portray an area of engagement. Each system also channels all user input through a graphical user interface (GUI). None was intended to be used with a natural language interface. However we have demonstrated through implementation that a natural language interface can be added to both the CIP and KOALAS. Although a natural language interface has not been implemented for TABS, it is clear that such an interface could be added.

The three systems are also similar in the role they play as decision aids. Each system provides information for people who themselves must reason about what decisions to make.

For the most part,[1] the systems do not reason about or advise people what to think or what to do.[2] Deciding what to think or what action to take are matters that are left up to the user. These systems aid their users by providing information that is useful and timely. This way of characterizing the person/machine interaction emphasizes the mind of the person. How useful a decision aid is at a given time depends on what the person is thinking and what decision he faces at that time.

Emphasizing the mind of the person is a useful heuristic that we will employ throughout this paper. Of the many things a person might be thinking we will be primarily interested in his focus of attention, i.e. what he is thinking about at a given time. As a working hypothesis, we will assume that the better a system provides the required information about what the person is focused on, the better that system will aid the decision.

The CIP, KOALAS, and TABS all respond to focus of attention in one very important way, by presenting a map. By looking at the map a knowledgeable user can rapidly focus on one or more of a large number of objects. Perceptual scanning is rapid and precise (to a degree). So by presenting a map a system offers a large amount of information from which a user can often quickly retrieve what will be useful at that time. Here the user performs the retrieval, but the system presents information in a form that can be rapidly retrieved.

Maps work for some of the large amounts of information these systems store. However, much important information cannot be easily shown on a map or retrieved by scanning through menus and mousing operations on the decision aids' GUI systems. We will present examples that show how some interactions with the systems can be achieved more easily and in more timely fashion through a natural language interface. We will illustrate the fact that natural language is useful because it closely tracks the focus of attention of a user. Given a natural language interface, a user can more easily communicate what he has in mind so that the system can more easily provide a helpful, relevant, and well focused response.

This paper is divided into the following sections. In Sections 2.0 and 3.0, we describe the Combat Information Processor and KOALAS. In Section 4.0 we describe a representation scheme for discourse that will allow focus of attention to be tracked by the interface. In Section 5.0, we describe how this representation scheme could improve the performance of a system that forecasts hypothetical battle situations. Section 6.0 describes our implementations and section 7.0 reviews our conclusions.

2.0 The Combat Information Processor

The Combat Information Processor, known as the CIP, was created by the Information Processing Branch at the Army Research Laboratory [Gurney 1991]. Its purpose is to gather, store, and display battlefield information. Figure 1 shows one of the CIP displays copied from a color monitor. Information on this map is encoded by colors and by the shapes of various icons. Lakes are blue, forests are green, and roads are red. Small boxes and other symbols represent sensors, brigades, companies, and other things that move. By simply looking at the displayed map one can learn a good deal about an evolving combat situation, because much of what the CIP has to offer to a knowledgeable user is there to seen. For the rest, there is the menu driven graphical user interface, the GUI. One of the GUI menus appears on the right in Figure 1. The box along the bottom is the place for the questions and answers that the experi-

---

1. KOALAS does include a specialized advice generator along with information, but it is ultimately the operator's decision to accept the system-generated advice or not.

2. These systems differ from a system like RACHEL [Holtzman 1989], which calculates expected utilities in order to recommend choice of medical treatment to doctors and patients.

FIGURE 1. Combat Information Processor

mental natural language interface processes. (This box is not part of the original CIP. So when we refer to the CIP we will generally mean the CIP without the natural language interface.)

In the natural language Q and A box a question about the roles of some of the companies has been answered:

(1) *ENTER A QUESTION / What are the roles of the companies that are not armor units?*

ANSWER / infantry motorized_infantry

It happens that this question asks for information that is already available to perception; the roles and echelons of units are encoded in the various rectangular icons on the map. But the relevant information is spatially distributed over the map. So the person wanting an answer would have to visually scan this visible data base while performing a tally in order to

assemble an answer. The natural language interface, on the other hand, leaves the searching and computing to the machine so that the person can keep his attention focused on his reason for asking the question.

Visual access to the map will not supply all of the information a person may need. Consider facts about the past, how fast something is moving, or the location of something to the nearest meter; none of these things could be easily shown in a single picture. At the scale of the CIP map the motions of some objects will not be perceptible. And spatial resolution is also limited. Even if the required resolution were available (perhaps by zooming in on one region of the map) humans cannot read off distances between objects that they perceive.

The CIP stores vast amounts of information that is not presented on the map or is not easily retrieved from the map. A useful way to think about how a person will want to use the CIP to extract this information is in terms of the person's mental focus of attention. A person uses the CIP map by focusing his attention on one symbol or group of symbols at a time, perhaps a bridge, or the line of sight from a sensor to a choke point. Likewise that person will want to use the vast non visible CIP database by focusing his attention on one object, property or relation, or groups (or other logical combinations) of these at a time. The CIP (without the natural language interface) channels all user interaction with the database through its GUI. This GUI provides an array of menus and means for pointing and drawing on the map. We will use one of the pop-up boxes as our example.

The CIP GUI was designed to use a large number of application modules. Underlying each possible GUI interaction there is a module that runs a procedure. One of these procedures displays text in pop-up boxes that attach to icons on the map. If a person points at (an icon for) a unit, a box will appear that writes out the echelon, the coordinates, and the time of last reporting for that unit. So in this context, pointing at the icon is semantically equivalent to asking the three questions (2-4).

(2) What is this?

(3) What are this thing's coordinates?

(4) When was this thing last reported on?

It turns out that if we were to examine the range of possible CIP GUI moves and menu choices we would find that most of them could be translated into semantically equivalent English sentences – assertions, commands, or questions. This means that we should be able to construct a natural language interface that can do much of what the given GUI can do. We will also show that, at least in the case of the CIP, a natural language interface can do a lot more than the GUI alone, that it can give a user more control over his system.

The natural language interface we implemented for the CIP processes questions like the following. Suppose there is only one brigade in the CIP world. Then a person could enter (5) and get back the same answer that would appear in a pop-up box (*if* he were pointing at the right icon).

(5) Where is the brigade?

We now have two ways to extract the same piece of information, and considerations of ease of use could come into the choice of method as well as the decision whether building a natural language interface would be worth the trouble.

But the pop-up box only answers three virtual questions, whereas a person might think of all sorts of questions about the CIP world. In fact the appearance of the box could suggest other more interesting questions. After learning how to use the CIP GUI a person will want to ask other questions such as those in (6), as well as the question in Figure 1.

(6) Where are the enemy units?

Are any of the enemy units not moving?

Are there any sensors that are above 1300 meters?

What are the green (map) features?

These questions focus attention on: the set of all enemy units, the property of motion, a particular elevation, a particular color.

The natural language interface can process these questions, whereas at the present time the GUI can process none of them – either virtually or otherwise. Answering these questions requires a natural language interpreter that can gather database items into sets (like the set of all enemy units), perform logical operations (like negation), as well as handle various modifiers (including relative clauses like 'that are above 1300 meters'). These and other linguistic constructions give natural languages their expressive power.

We could, of course, add menu choices to the GUI to process some of these questions. But without creating a recursive language, like English, this would be a losing game resulting in a proliferation of menu choices, where menus have sub-menus that themselves have sub-menus and so on. One advantage of a natural language interface is that it permits a person to bypass many menu operations by typing in one well formed sentence. It saves labor; it reduces cognitive effort. A person can become confused and overloaded with irrelevant information if the machine (or other person) he is communicating with interferes with his focus of attention.

3.0 KOALAS

The KOALAS Test Planning Tool [Barrett et al. 1990] is a demonstration military system design tool developed by Los Alamos National Laboratory for the Naval Air Systems Command. It uses a simulation to depict engagements between friendly Naval and hostile aircraft on a synthetic AEW radar screen, and provides a set of text windows presenting current fighter and threat status, communications transactions, and sensor readings. Although the system contains an internal rule-based tactical advice generator, the human operator has the final decision over all fighter actions and threat hypotheses during each simulation run. Indeed, one application envisioned for the KOALAS tool by its designers is to train personnel to improve their decision making during tactical engagements.

In the original graphical interface to KOALAS (Figure 2), pop-up menus with selections sensitive to mouse-clicks were the only means for the operator to enter commands. In many cases these menu selections resemble the verb-argument structure of natural language sentences, so building a natural language interface that integrated easily into the graphical interface was straightforward. This integrated interface that NRL developed for KOALAS is called Eucalyptus [Wauchope 1992]. One design decision in Eucalyptus was to correlate natural language imperatives one-to-one with the existing menu and dialog box structure whenever possible. For example, consider the command

(7) Move fighter 1 to station 3.

This type of command replaces a series of manipulations of menus and mouse-clicks that can be complicated; so Eucalyptus saves labor and makes the person/machine interface more natural and easier on the mind. The benefits here are similar to those we found in the CIP interface. Eucalyptus also includes a natural language query facility that permits the user to quickly access specific information in his focus of interest without having to search for it in the KOALAS scrolling graphical display windows.

Eucalyptus differs in one important way from the CIP natural language interface. The later was designed to bypass the menu system, offering a person choices to ask questions that were not available in the menus. Eucalyptus, on the other hand, assumes that the KOALAS graphical interface represents all the data access and command functionality that the system's designers intended for it to provide and support. Natural language then represents a more nat-

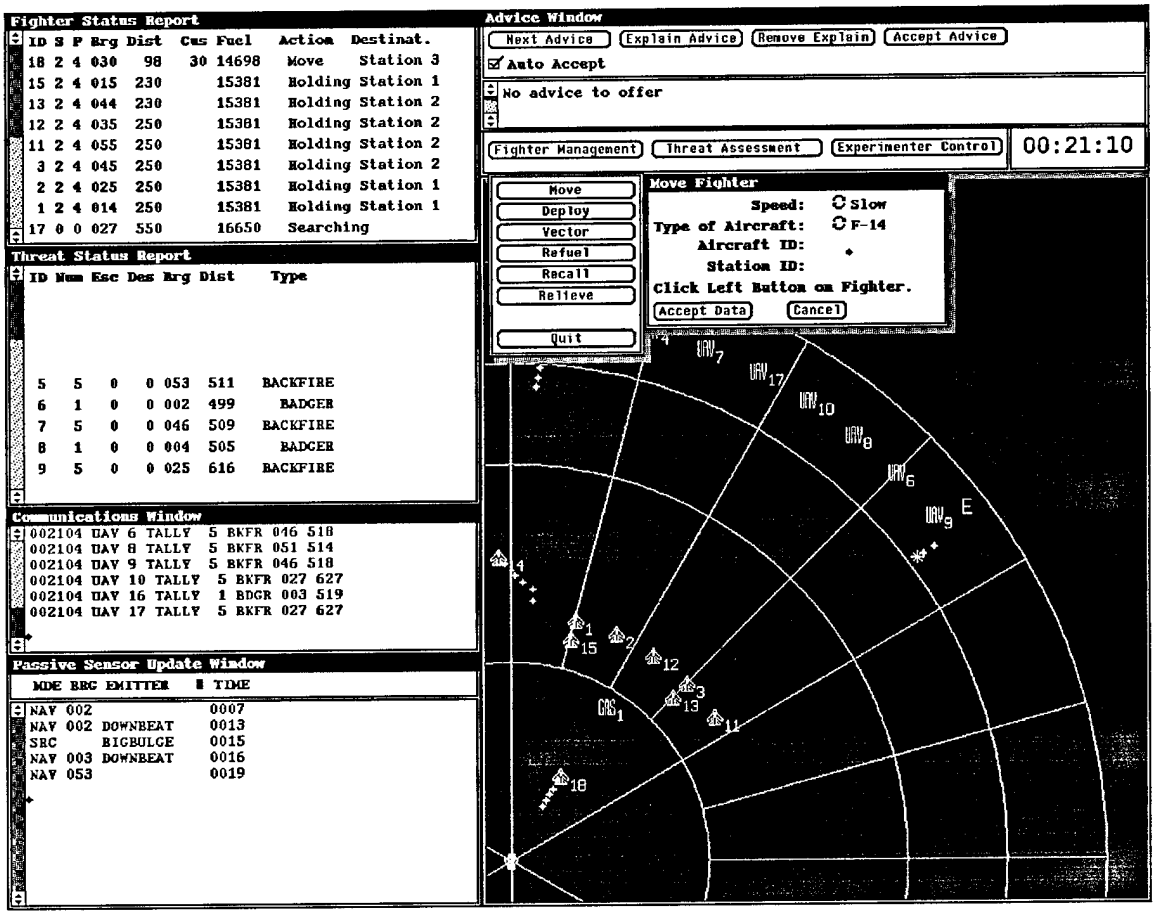

FIGURE 2. KOALAS

ural and convenient way to access that predetermined functionality than the graphical interface provides. For example, a Eucalyptus query like

(8) Which F14s and UAVs are not holding station?

uses natural language quantification, negation and logical connectives to quickly and concisely obtain the specific information being focused on by the user, who would otherwise have to scroll around in the graphical display window, mentally constructing the appropriate set of objects from the individual aircraft status reports.

In regard to commands, Eucalyptus emphasizes choice of input medium and modality. Thus a user can enter the same command by typing it, by constructing it from choices in a menu box, or by speaking into a microphone. In addition, Eucalyptus accepts multimodal interactions. Normally a KOALAS command must be fully specified, as in (7) above, which conveys the actor, the type of action, and the destination. Given the option of using natural language, a person can issue commands that are grammatically correct though underspecified as, for example,

(9) Move a fighter.

There are many possible reasons why a person would say (9) rather than a fully specified command; as a trainee, he may not know a destination is wanted, or he may think KOALAS will assume a destination that seems obvious, or he may be so concentrated on launching the fighter that the destination is not in mind. Whatever the reason, his use of (9) conforms to his cur-

rent focus of attention. In this way Eucalyptus is properly flexible; it allows the person to communicate *something*. Eucalyptus will process this partial thought and will respond by prompting for a destination. Finally, the user can mix natural language and pointing gestures, as in

(10) Move this fighter out here.

allowing a natural focus on the spatial representation image offered by the radar display and freeing the user from having to retain the sector number of the destination ("station 3") in his memory.

Even though Eucalyptus tends to map imperative orders one-to-one to the commands available from KOALAS menus, it allows much greater flexibility in the ways a person can refer to the fighters, destinations, and other objects on the radar screen. Having entered (7) the trainee could later send a second fighter to the same destination by saying (9).

(11) Move a fighter to the same station fighter 1 is moving to.

Even though the meaning of (11) is clear, the KOALAS GUI offers no way to enter this command. The command box requires that every destination be specifically identified either by mouse click on the radar screen, or by name. This is the kind of rigidity that natural language interfaces can easily overcome. In this case Eucalyptus processes not only what objects the person has in focus; it also handles the way he is thinking about those objects.

4.0 Anaphora and Discourse Representation Theory

We have been motivating NLIs primarily by pointing to the importance of focus of attention in the mind of a person who wants to use a machine to help make a decision. We have recommended that a person be allowed to choose his means of expression – which choice will be governed by his mental state – which state employs focus of attention. In all of the above examples of person/machine dialog we can see the implicit influence of focus of attention. What is focused on helps determine what input sentence that person will construct. In these examples, it was not necessary for the NL processor itself to represent or keep track of focus of attention. The NLI could process each sentence in isolation without regard to anything the user may have said (or entered) previously.

In this section we will motivate and present a NL system that processes sequences of sentences that form cohesive discourses. In order to process these discourses, the system must represent and reason about what the focus of attention is at a given time. We will first consider focus of attention as it arises in natural language discourse involving anaphora, i.e. interpretation of pronoun reference.

For our present purpose we define a discourse as two or more sentences (or sentential fragments) that are connected in some way that matters for the interpretation of what is said. Here is an example of the kind of short discourse that might occur in a session with Eucalyptus (12-13).

(12) Move a fighter to station 2.

(13) Tell me what bearing it has.

Sentences (12-13) form a discourse because the interpretation of the second sentence depends on fixing the referent of the anaphor, i.e. *it*, which in turn depends on the interpretation of the first sentence. In general, discourses form cohesive wholes. The cohesion in this example comes from focusing attention on that fighter. Whoever says (12) introduces a fighter into his thoughts. When he goes on to say (13) he is still focused on that same fighter. And whoever is following these orders better be able to get the focusing right in *his* thoughts. Thus, we have here a case where tracking focus of attention is essential to understanding what is said.

People generally find it easy to track the focus of what is being said in natural language discourse. However, implementing discourse processing within the context of human-machine interfaces is not a trivial matter. There are research problems involved in modeling discourses. There are simple discourses that are easy to model and there are more interesting ones that are not yet well understood. In what follows we will concentrate on discourses that can be modeled within the current state of the art.

A natural language interface that can process anaphora will allow the user to express thoughts cohesively as in (12-13). This will make the interface more user friendly. Also, sometimes people use pronouns because they cannot think of any other good way to say what they have in mind. In these cases it will be essential for the interface to process discourse. The following situation is an example where a person may believe a pronoun is necessary.

Assume that there are two fighters heading for station 2 – one launched by the user when he said (12), the other launched some time earlier. In this case, discourse (12-13) still conveys what he has in mind because *it* in (13) still refers to the fighter mentioned in (12). Here the user thinks he must use a pronoun in (13); he would not try to say (14) because he knows there is no single such fighter as the words *the fighter* would suggest.

(14) Tell me what bearing the fighter moving to station 2 has.

In this case, (13) with a pronoun conveys what he intended, while (14) would not.

This leaves the task of processing simple anaphoric pronouns as they occur in simple discourses as a good place to begin upgrading our natural language interfaces.[1] We have moved to needing an NLI that can explicitly represent and calculate – as a minimum – just what is in focus in these cases. After dealing with anaphora we will show how the model for discourse processing can be extended to handle some related linguistic phenomena that occur in TABS.

Some discourse processing has been incorporated into the CIP NL interface and into Eucalyptus. However, rather than discuss these implementations, we will now (and for the rest of the paper) discuss a comprehensive model for discourse processing currently under study at NRL. The model we present here is based on Discourse Representation Theory (DRT) [Kamp 1981, Heim 1982]. DRT has recently been under development and study by linguists, computational linguists, and philosophers [Roberts 1989, Covington 1988, Johnson 1986, Smith 1987]. We will review some of the details of DRT, so that we can show how DRT models focus of attention.

DRT can be implemented in the form of an algorithm that takes the parsed logical forms of individual sentences as input. This algorithm creates a representation of a discourse (i.e. the discourse representation structure) that stands between the individual English sentences and the database (or knowledge base) of the decision aid that the person is interacting with. This important innovation was originally designed to handle some well known linguistic problems.[2] DRT is in fact an elegant (that is, simple) formalism for representing the kinds of discourse we are interested in.

Consider the following discourse of three sentences that might be used to update a CIP type database (15).

(15) Unit 253 is a brigade

It is a friendly unit

---

1. There is more to discourse processing than handling anaphora. Some of this we will take up in section 5.0 below. Much of the rest we will not discuss here.

2. One of these is the problem of processing so called donkey sentences such as "Every farmer who owns a donkey beats it."

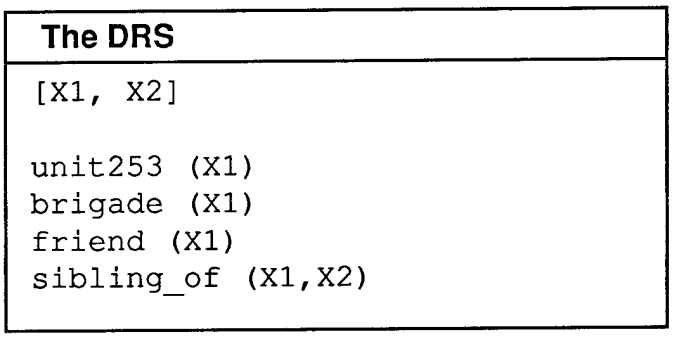

FIGURE 3. Assertion Discourse

And it has a sibling.

This discourse produces the DRT representation shown in Figure 3. This representation will be called the discourse representation structure – DRS.

This DRS (in Figure 3) consists of a list of discourse entities `[X1, X2]`, followed by a set of conditions, `unit253(X1)`, `sibling_of(X1,X2)`, etc. The list of discourse entities represents what is in focus and the conditions represent how these entities are thought about. While building the DRS the DRT construction algorithm computes the significance of any linguistic discourse devices. In this case the discourse devices are the two uses of the anaphoric pronoun – it. The rules for building a DRS represent the effect of such anaphors by variable binding. In our case the referent of 'it' was judged to be unit 253 and so the algorithm used the variable X1 (which had already been chosen to stand for unit 253) for the first argument of the condition `sibling_of(X1,X2)`.

Looking at the DRS it is not apparent how many sentences were input, nor how the pronouns were used. The discourse has a "mental" structure of its own that may cross sentence boundaries and that could have been expressed through alternative combinations of names and pronouns. The order in which objects are mentioned is important however and this order is preserved in the list of discourse entities. Also it appears that the DRS of this kind of discourse is determined completely by the syntax of the original sentences. What we have here is someone keeping unit 253 in focus as he elaborates on what he knows about that unit. Very simple sentences signal that he is elaborating in this way. So the DRT algorithm can track his focus of attention. As we said above, this tracking is easy for humans. In this case it is easy for our natural language processor as well.

Some discourses create complex focus spaces. In these cases, one focus space will be contained within another space. The two sentences (16) and (17) form such a complex discourse.

(16) Every company has a sibling

(17) Unit253 has an amphibious sibling.

The DRS for (16-17) appears in Figure 4. The content of (16) is equivalent to *if something is a unit it has a sibling*. This syntax creates a sub discourse marked off by the two interior boxes that are connected by the horseshoe in Figure 4. An important rule of DRT is that discourse entities introduced within a sub DRS cannot bind with (i.e., be referred to by) discourse entities outside that sub DRS. Therefore, X3 – which is outside the sub DRS – cannot bind with X1 or X2.

Sub DRS's such as the one above are triggered by syntactic structures of English such as universal quantifiers; in our case 'every unit'. By constructing these sub DRSs, DRT helps explain certain linguistic phenomena – such as when an anaphor (a pronoun) can and cannot

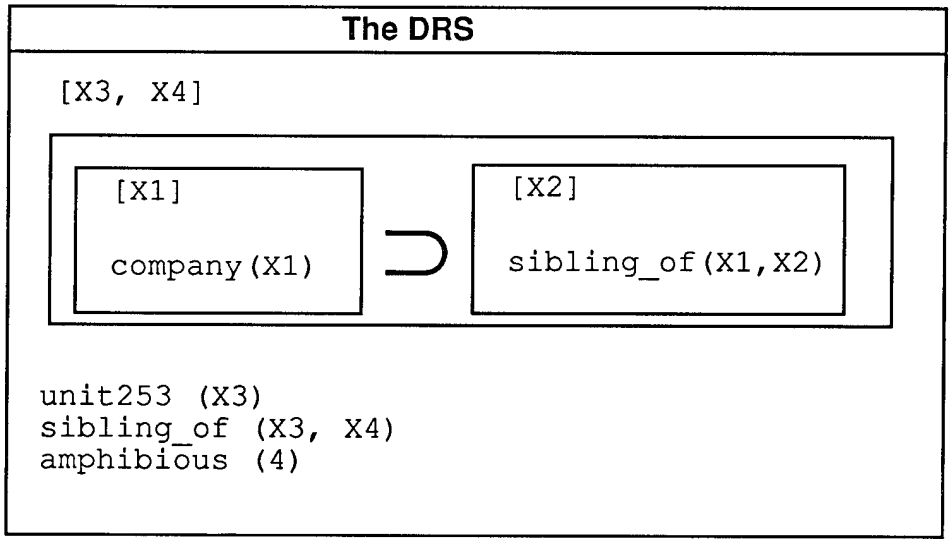

FIGURE 4. Complex Discourse

refer to a previously mentioned entity. Thus, in solving these linguistic problems, DRT also accounts for the complexity of focus of attention. In the above discourse it's not just a matter of whether an entity is in or out of focus. Three entities are in some sense in focus: a company, a sibling, and a unit. But these three entities are in different focus spaces, and thus cannot co-refer.

In the preceding example we see that the focus spaces can proliferate as sub DRSs. We will call the sub space in the above example a conditional sub space because it models the "IF P THEN Q" construction in English. There will have to be several other types of sub space according to DRT. One of these is the DOIT sub space, which will be required to model the KO-ALAS discourse about moving a fighter (12-13 above). The DRS appears in Figure 5.

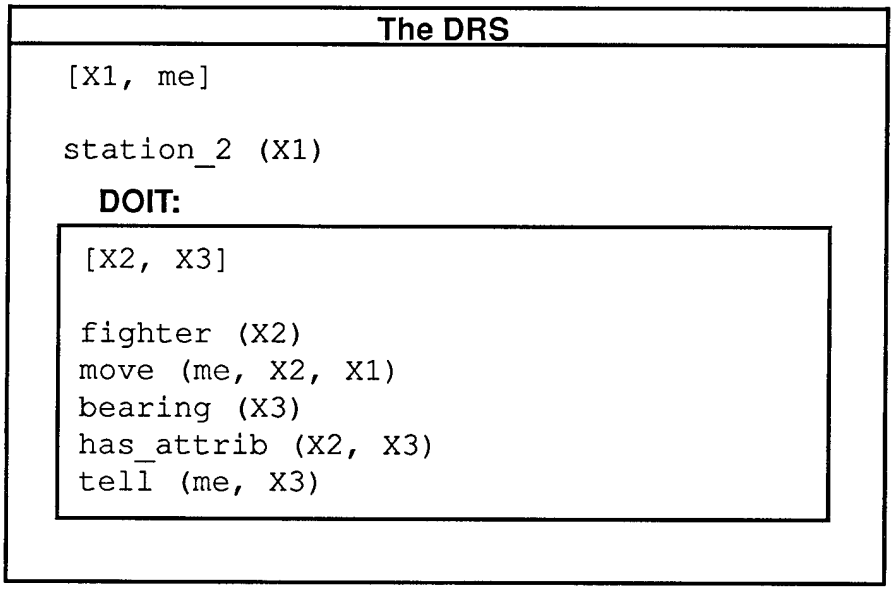

FIGURE 5. Command Discourse

The structure of this DRS is different from the previous one. Both (12) and (13) are commands; that means DOIT sub spaces must be created; however, here we create only one DOIT subspace. The reason is that (13) continues the command (imperative mood) begun by (12). A theory of modal subordination [Roberts 1989] has been developed to explain just when new sentences should be added to existing spaces and when they should create new spaces. Given that the content of (13) belongs in the existing DOIT space, there is no problem having *it* refer to the fighter. However, if the user went on to say (18), the listener (or machine) might have a problem answering – probably because the fighter may not have been chosen yet.

(18) What is its course?

DRT explains this because the question does not continue the imperative mood of the previous sentence. Instead (18) creates a new WH space which will be outside the DOIT sub space. Therefore *its* cannot refer to entities that are inside the DOIT sub space.

Although these DRSs can be somewhat complex, the ins and outs of these spaces and sub spaces are, of course, easily handled by people when they use English to communicate. That is to say two people can easily keep track of what things are in focus and how they are in focus. They do it because they know how to process anaphors, quantifiers, syntax that indicates the mood of a sentence, and many other linguistic constructions. They do not, however, explicitly represent these DRSs to themselves. Hence we would probably not want to show the DRSs that our interface creates to a user, as that would be confusing. However, our NLI must construct the DRSs to process sentences while keeping track of the user's focus of attention.

We have seen that some instances of the required focus tracking can be processed by an interface that employs DRSs according to Discourse Representation Theory. We will show in the next section how this theory can also be used to model thinking about hypothetical situations and alternative realities.

5.0 TABS

TABS is an antisubmarine battle management system. Its purpose is to forecast and display positions and tactical states of enemy submarines. Like the CIP and KOALAS, it was designed and built with a GUI as the only channel of communication from user to machine. A typical display screen appears in Figure 6. In this figure the locations of two submarines – an Oscar and a Victor-III – are shown as probability density grids. The probabilities of four possible tactical states are also shown for each of these two threats as overlays on the map. The states are **fleeing, attack, in localization**, and **out of contact**. The BG box is the Navy battle group.

In some respects TABS is a Naval version of the CIP. It shows where everyone is on a map; it shows other, non visual, descriptive information in boxes that overlay the map; and it provides an elaborate GUI for the user to query and manipulate the system in various ways. However, TABS can do more than display information about the actual world – what TABS calls the real time tactical picture. It can also forecast what the future will look like. For example, if the user requests a forecast at real time plus eight hours, the display will show that the battle group has moved to a new location and that estimates of the tactical states of the two submarine threats have changed.

In addition to forecasting the actual future, TABS can also generate a what-if picture. This is a picture of what the future would look like if the present were different than it is. For example, it could display a what-if tactical picture projected eight hours into the future in which, contrary to fact, the two threats are both Oscars, rather than an Oscar and a Victor-III.

FIGURE 6. TABS

The what-if picture is generated by the user dragging icons around from one area to another on the screen and by making a few selections from GUI menus. He first makes TABS build a starting what-if picture. This looks exactly the same as the real picture but it is a duplicate; from now on changes in the what-if picture will cause no changes in the real. Next he changes the Victor-III to an Oscar in this what-if picture. At this point the what-if picture differs from the real picture but the time is still the present time (real time). Finally the user makes TABS forecast this real time what-if picture eight hours into the future. We call this a picture of a counterfactual situation.[1]

Let us now consider how NL might play a role in the TABS what-if capability. Having built the what-if picture it is as if the user had either asked a question (19) or issued a command (20).

(19) What would the tactical picture be if the Victor-III were an Oscar and the time were eight hours from now.

(20) Show me the tactical picture eight hours from now if the Victor-III were an Oscar.

---

1. A likely reason for this particular exercise is the possibility that the Victor-II is really an Oscar.

The effects of all of the various manipulations and operations the user performed within the GUI to generate the what-if picture can therefore be captured in a single sentence. As with the NLI for CIP, a NLI for TABS could process these kinds of questions and commands directly. So the TABS NLI would also assist the user in focusing his attention where he needs it, as a decision aid should. Here we have a user focusing his attention on a submarine and a tactical picture.

There is an important difference between focus of attention with the CIP and focus of attention with TABS. The CIP user focuses on and asks questions about real objects and relations that are represented in the CIP databases. Here focus of attention amounts to making a selection from the database. The TABS user, however, can use the what-if capability to focus his attention on objects that do not exist and on situations that are not real. (Obviously, such objects are not represented in the TABS database.) If he wants to use NL to convey these thoughts, he may use sentences like (19-20) above. Using DRSs, a natural language system can process such sentences about non-real objects, because the DRSs that the discourse processor would construct for these sentences would be separate from the TABS database.

We call (19-20) counterfactual sentences because they convey thoughts about non-factual situations. The use of the subjunctive verbs *would* and *were* make it clear that counterfactual, non real situations are in consideration. The CIP and KOALAS NLIs do not currently process counterfactual sentences[1] because they cannot distinguish between what is real and what is only thought about.[2]

A DRT NLI (DNLI) that processes counterfactuals can offer several benefits.

First, a NLI that processes counterfactuals can give a user more flexibility to express himself as he chooses. The phrasing in (19) is somewhat complicated and perhaps a little confusing. If so, that may be because (19) does not track the flow of focus of attention through time. In (21) the focus is first on the Oscar, next on the tactical picture.

(21) Assume the Victor-III is an Oscar.

What would the tactical picture look like at plus eight hours?

(21) does the same job as (19) and any DNLI that can process the one can process the other. In fact the DRSs for (19) and (21) would be equivalent.

Second, with a DNLI, the user can direct TABS to present information that is more responsive to his needs at a given time. On the other hand, the GUI for TABS presents information in rigid formats using icons and boxes that overlay its map (in this it resembles the CIP and KOALAS). With a DNLI, the following question (22) allows a user to quickly focus on the question of tactical contact without having to first ask for a what-if map and then read through the overlays on the map for an answer.

(22) If the Victor were an Oscar would it be in contact?

Third, using a DNLI could make the system interface more transparent. A user can think about and talk about the domain exclusively without referring to the maps, icons, and overlays that make up the system interface. For example, perhaps the user would prefer to enter (22) so that he can keep the real tactical picture in the display while he wonders about how one factor (of very many) would change. (22) combines counterfactual thinking with focusing

---

1. We could easily enhance the CIP NL and Eucalyptus system parsers to handle subjunctive verb forms, but that would not solve the problem of meaning. We would need a different type of underlying representation for what the sentences mean. This is what DRT can provide. See [Goodman 1947, Kratzer 1977, 1981] for research issues in processing counterfactual sentences.

2. KOALAS makes extensive use of hypothetical objects to represent possible incoming enemy raids, but these entities are stored in the database and treated by the interface as pseudo-objects explicitly referenced by name. Counterfactuals could provide a more natural interaction with this facility.

on one aspect of an alternative reality. Note also that (22) refers to domain objects only while in (19-21), we have someone referring to domain objects along with the tactical picture which is a TABS system object.

Fourth, in a DNLI with counterfactuals, it is possible for the user to straightforwardly gather and organize information to supply an answer to a question, as in (23).

(23) If the Victor-III or the Oscar were a Delta could we be under attack within the next eight hours?

To answer (23) using the TABS GUI the user would have to first create one what-if picture with the Victor-III changed to an Oscar and then project this picture to a series of future times up to plus eight hours. Then he would have to return to the real time picture and repeat these moves for the (real) Oscar making note of any relevant results along the way. Of course TABS itself would have to perform a complex series of calculations to generate enough information to answer (18). The point here is that we may not want the user to walk through all these steps. We may simply want to show the answer. This is a striking advantage of the DNLI over the GUI.

Note that TABS can provide all of the information needed to answer (19-23) above. The additional processing required to activate the TABS procedures and combine results to deduce answers comes with the NLI. We are, in effect, getting better performance out of a system by adding general purpose language processing capabilities.

Let us now consider how to represent real vs. counterfactual (non-real) situations in DRT. Counterfactual situations must be represented separately from real situations. We will take (22) as an example to be processed using DRT. A counterfactual is a special type of conditional sentence. We use DRT sub spaces for this purpose, as shown in Figure 7. This DRS bears a superficial resemblance to the sub DRS for (24) shown in Figure 8.

(24) Is every Oscar in contact?

(24) means the same as (25):

(25) Is it true that if something is an Oscar it is in contact.

The differences between Figure 7 and Figure 8 are important. They represent differences between ordinary conditionals, if P then Q, and counterfactual conditionals, if P were the case then Q would be the case. The counterfactuals require that there be a necessary connection from P to Q. This connection is symbolized by the arrow connecting the two boxes in Figure 7. The ordinary conditionals require only that as a matter of fact whenever P is the case Q is the case. Thus (25) asks only whether every Oscar happens to be in contact, not whether

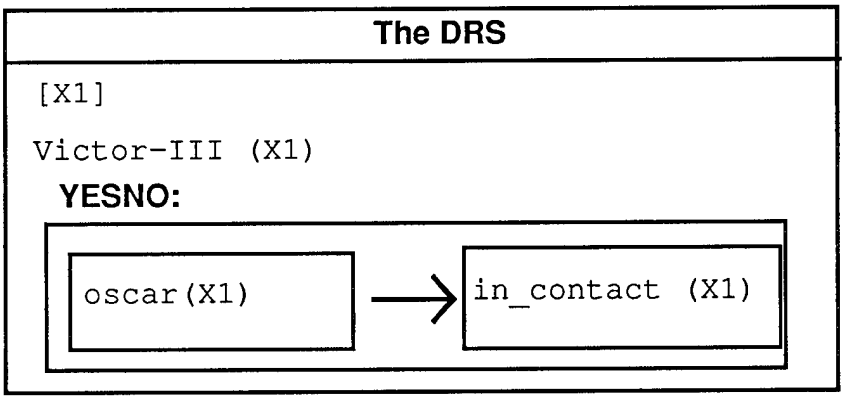

FIGURE 7. Counterfactual Query Discourse

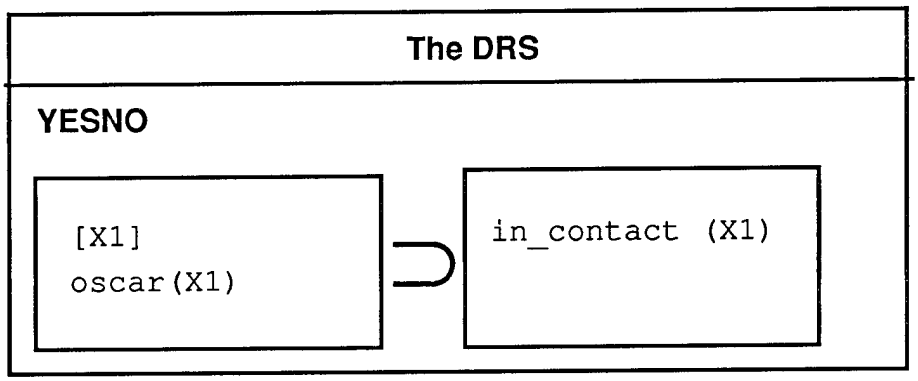

FIGURE 8. Conditional Query Discourse

they must all be in contact. (22) is asking for a necessary connection. Therefore the only way to answer (22) in the affirmative is to deduce contact from Oscarhood. Thinking about alternative realities involves thinking about necessary connections. If a Victor-III changes to an Oscar in a what-if picture, TABS looks up the essential (that is necessary) properties for an Oscar and inserts them into the new picture. Likewise, if a user submitted the question (22) through an NLI, TABS would try to deduce that the Oscar must be in contact. If it succeeded, it would answer yes to the user's question.

DRT must also separate counterfactuality from reality. The two boxes joined by the arrow in Figure 7 form a sub space that is not accessible from the outer space which represents reality. However, sentence (26), the continuation of the (22) discourse illustrates that modal subordination should still work.

(26) Would it be attacking our group?

Here 'it' should refer to the counterfactual Oscar (not the real Victor-III) while 'our group' should refer to the real battle group. In examples like (26), a person can express a thought that combines reality and unreality without confusion. The DRS in Figure 9 represents these connections appropriately.

6.0 Implementation

The CIP is written in C and runs on Sun workstations, as well as a custom platform. The natural language interface to the CIP processes questions such as the examples in section 2.0. This module runs as a separate process on the same workstation as the CIP. The parser and interpreter were written in Prolog using a slot grammar [McCord 1987]. In order to answer a question, the natural language interpreter directly accesses the CIP data from memory.

KOALAS is written in C and runs on Sun-4 workstations. It is linked into a single process with the natural language processing component, which is written in Common Lisp. The PROTEUS parser uses an augmented context-free phrase structure grammar written in Restriction Language [Grishman 1986]. Other modules convert the parse to quantified logical form, which when evaluated accesses KOALAS data through C language procedures based closely on those that underly the graphical user interface.

The discourse processor discussed in section 4.0 was written in Prolog. It uses a unification grammar [Covington 1989, Shieber 1986] to translate English sentences into logical forms. A system of rules [Johnson 1986, Covington 1988] embeds these logical forms into the

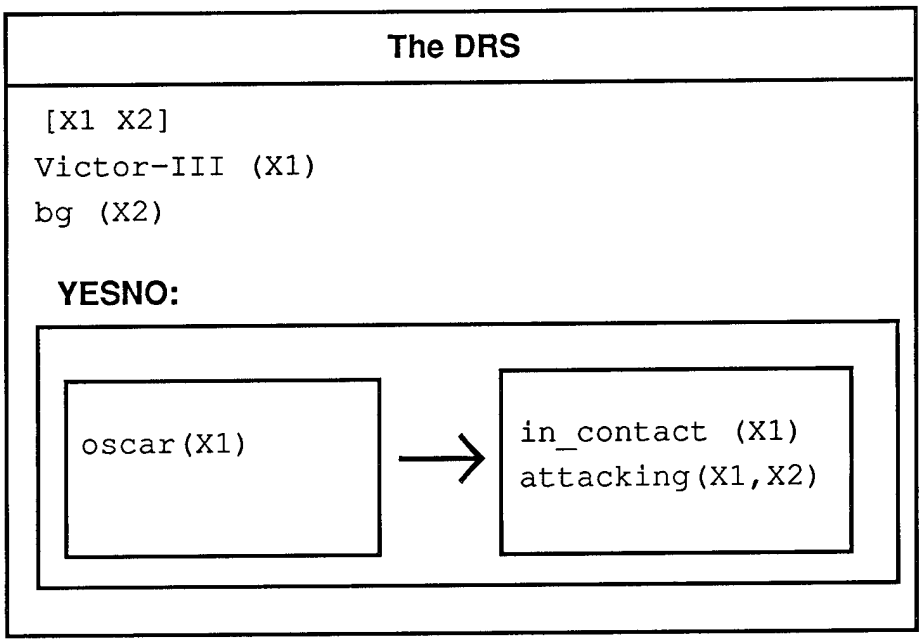

FIGURE 9. Counterfactual Modal Subordination

discourse representation structures. This system is under development and has not been interfaced with other systems at this time.

7.0 Conclusions

In this paper we have described three decision support systems with graphical user interfaces. We have illustrated that interactions with such systems can be improved with the addition of natural language interfaces. NLIs can be added to systems after the initial development phase without loss of functionality. Furthermore we have shown that a hybrid natural language and graphical user interface with multi-modal capability improves access to decision aids more than either one independently. We have also shown that natural language interfaces can help a user maintain his focus of attention in complex situations.

A natural language interface to a decision support system has a number of important advantages. A natural language interface is especially useful in recognizing what the user is thinking about at the time of interaction, i.e. identifying the user's focus of attention. It provides a way for the user to maintain focus on his problem without distraction. A natural language interpreter can access the system for the user and assemble a relevant, well-focused response. A NLI allows the user flexibility of expression. This is important if the user does not know how to say what he intends in the GUI format. NLIs can handle complex focus spaces since focus is readily conveyed through natural language. In fact, natural language may be the *only* way it can be readily handled. Finally, and importantly, natural language interfaces can handle a user's counterfactual thinking.

Because of the complexity and power anticipated in next generation decision support systems, natural language interface technology warrants a closer look. It can serve to help the user manage complex and quantitatively massive amounts of information in a natural and easily expressible way.

8.0 References

Chris Barrett and Charles Aldrich, KOALAS Test Planning Tool Concept Demonstration Users Manual, Los Alamos National Laboratory, New Mexico, 1990.

Michael A. Covington et al., From English to Prolog via Discourse Representation Theory, Research Report 01-0024, U. of Georgia, 1988.

Michael A. Covington and Nora Schmitz, An Implementation of Discourse Representation Theory, Research Report 01-0023, U. of Georgia, 1988.

Michael A. Covington, GULP 2.0: An Extension of Prolog for Unification-Based Grammar, Research Report AI-1989-01, U. of Georgia,1989.

Nelson Goodman, The Problem of Counterfactual Conditionals, J. of Philosophy, 1947.

Ralph Grishman, PROTEUS Parser Reference Manual, PROTEUS Project Memorandum #4, Courant Institute, New York University, 1986.

John Gurney, A Natural Language Interface Generator, for a Graphic Information System, Harry Diamond Laboratories, draft, 1991.

Irene Heim, The Semantics of Definite and Indefinite Noun Phrases, Dissertation, U. of Massachusetts, 1982.

Samuel Holtzman, Intelligent Decision Systems, Addison-Wesley, 1989.

Mark Johnson and Ewan Klein, Discourse, anaphora, and Parsing, Eleventh International Conference on Computational Linguistics, COLING86, Bonn, 1986.

Hans Kamp, A Theory of Truth and Semantic Representation,in J.A.G. Groenendijk et al, eds., Formal Methods in the Study of Language, Mathematisch Centrum, 1981.

James Kilgore, TABS [personal communication].

Angelica Kratzer, What 'Must' and 'Can' Must and Can Mean, Linguistics and Philosophy, 1977.

Angelica Kratzer, The Notional Category of Modality, in Hans-Jurgen Eikmeyer and Hannes Reiser, eds., Words, Worlds, and contexts, Berlin, 1981.

Elaine Marsh, Towards Friendlier User Interfaces for Expert Systems, Proceedings of the 1991 World Congress on Expert Systems, Pergamon Press, 1991.

Michael McCord, Natural Language Processing in Prolog, in Adrian Walker et al, eds., Knowledge Systems and Prolog, Addison-Wesley, 1987.

Craige Roberts, Modal Subordination and Pronominal Anaphora in Discourse, Linguistics and Philosophy, 1989.

Stuart Shieber, An Introduction to Unification-Based Approaches to Grammar, CSLI Lecture Notes No. 4, Stanford University, 1986.

Richard Spencer Smith, Semantics and Discourse Representation, Mind and Language, 1987.

Kenneth Wauchope, Eucalyptus: An Integrated Spoken Language/Graphical Interface for Human-Computer Dialogue, Proceedings of AVIOS '92 Voice I/O Systems Applications Conference, Minneapolis, MN, 1992.